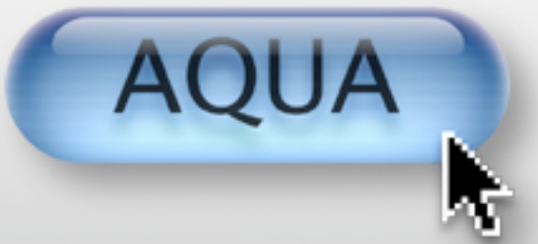
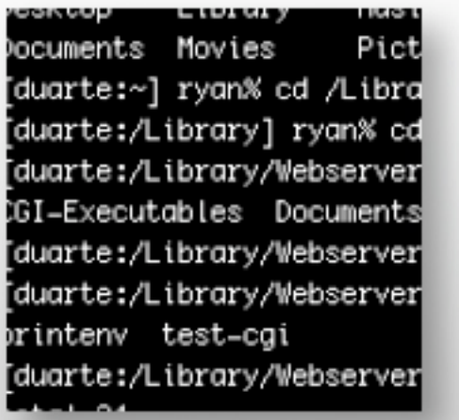
Unicode for Japanese, Chinese …and Everything Else

Session 404





Unicode for Japanese, Chinese …and Everything Else

Deborah Goldsmith
Manager, Fonts/Unicode Liaison
International Text Group
goldsmit@apple.com

Introduction

- Applications must support Unicode to be competitive in Japan and China
- Mac OS X provides the tools your application needs to have great Unicode support

What You'll Learn

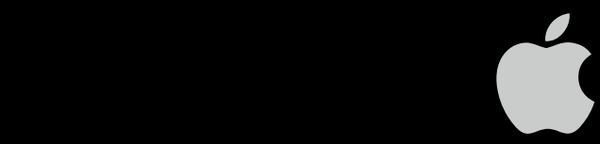
- New requirements from customers and governments in Japan and China
- How only Unicode meets these requirements, and more
- Great new Unicode–only features in Panther
- How Unicode differs from older approaches
- What you need to do in your application

The Problem

- Customers are demanding more characters
  - Variant Han characters for names (”Smythe” vs. “Smith”)
    - 渡辺 渡邊 渡邊 渡邊 渡邊
- Governments are specifying more characters
  - Japan: JIS X 0213
  - China: GB 18030
  - Hong Kong: HK-SCS
  - Taiwan: Big5E
- WorldScript cannot support any more characters

Unicode to the Rescue!

- Worldwide, cross-platform standard
  - Unicode Consortium, ISO/IEC 10646
- One unique encoding for all languages
  - Doesn't depend on font
  - No more mojibake: Δƒøˆ «√∑Ø±◊¿Œ
- You want characters? It has characters! 파워 플러그인
  - Unicode 4.0 has 96,248 graphic characters
  - Covers JIS X 0213, GB 18030, HK-SCS, Big5E, …



Mac OS X Unicode Coverage

- Lucida Grande
  - Extended Roman
  - Extended Greek
  - Cyrillic
  - Thai
  - Hebrew
  - Symbols
- Times, Helvetica, etc.
  - Extended Roman

Mac OS X Unicode Coverage

Great Japanese support!

- Hiragino: 6 DTP quality OpenType Type 1 fonts
  - JIS X 0213
  - Adobe Japan 1-5
  - Shaken 78 Kanji
  - NLC Shape recommendations
    - 国語審議会表外漢字字体表

Mac OS X Unicode Coverage

Great Chinese support!

- GB 18030 (Simplified Chinese)
  - Five font faces
  - 32,000+ glyphs
  - CJK Ideographs, Extension A, Yi
- New: HK-SCS, Big5E (Traditional Chinese)
  - 22,000+ glyphs

Mac OS X Unicode Coverage

More languages!

- New: Extended Arabic
- Hebrew
- Thai
- Cyrillic
- Devanagari, Gurmukhi, Gujarati
- New: Unified Canadian Aboriginal Syllabics, Cherokee
- New: Unicode Symbol blocks
  - Misc. Symbols, LetterLike Symbols, OCR, Currency, Box Drawing, Block Elements, Geometric Shapes, Math Operators, Arrows, General Punctuation

Other International Improvements

- Unicode text drawing much faster!
- Improved bidirectional support
  - Heuristic base direction detection
  - Unicode 4.0
  - Many bug fixes
- Indic layout bug fixes
- Thai word break (10.2.4), now with user dictionary
- More languages selectable for third-party apps

But What About Hanunoo?!

- Step 1: Add a font
  - See http://developer.apple.com/fonts/ for documentation and tools
- Step 2: Add a keyboard or input method
  - See TN 2056 for keyboards
  - See “BasicInputMethod” sample code

New Text Input APIs

- Variant Glyph Access protocol (10.2)
  - Allows access to variant glyphs (Han, Roman, and more)
- New TSM protocol improves accuracy, supports new HI features for input methods
  - Kotoeri analyzes document for better conversion accuracy
- Extension to Carbon Font Panel protocol for extended text attributes, font features, multimaster

Input Menu Improvements

- New merged, streamlined input menu
  - IM modes appear individually (for revised IMs)
  - No more pencil menu
  - Older IMs supported transparently

Text Input Improvements

- Kotoeri has better accuracy, more flexible interface
- Hanin Input Method (10.2.4)
  - Much easier for Bopomofo, Pinyin
- ITABC now supports GB 18030
- More keyboards included for more languages

Demo

Yasuo Kida
Manager, OS Engineering Tokyo
Michael Grady
Software Engineer
International and Text Group

http://developer.apple.com/wwdc2003/urls.html

International API Improvements

- Locales, date/time/number formatting/parsing in CF:
  - `CFLocale`
  - `CFDateFormatter`
  - `CFNumberFormatter`
- Many more locales for collation, formatting
  - Based on open source ICU library
  - ICU not yet supported for direct access
- Collation up to 200% faster!

What Cocoa Apps Need to Do

- Use Cocoa text system
- Don’t subclass `NSSimpleHorizontalTypesetter`
  - Will disable bidi and advanced layout
  - Subclass `NSATSTypesetter` instead
- Be careful when you save attributed text in a document
  - Can lose variant glyphs, advanced text attributes

What Carbon Apps Need To Do

Easy version

- Use CFString for Unicode text storage
- Use HITextView or MLTE for editing, display of text
- Support Font Panel, including advanced attributes

What Carbon Apps Need To Do

Custom text engine version

- Use `CFString` or `UniChar` for Unicode text storage
- Use ATSUI for Unicode text drawing
- Use TSM for Unicode input via *CarbonEvents*
  - Support input method active area
  - Support new content access protocol
  - Support variant glyph input and storage
- Support Font Panel, including advanced attributes

Unicode: Quick Review

- Major differences from WorldScript approach
- Characters represent information (“spoken”)
- Glyphs are shapes on screen, paper (“written”)
- Often 1:1, but not always
  - Arabic, Indic, but even English and Japanese
- Unicode rendering must map between the two
  - ह ि न् द ी ⇒ हिन्दी
  - م ا ك ن ت و ش ⇒ ماكنتوش
  - Re´sume´ ⇒ Résumé

Unicode: Gotchas

- Surrogate pairs (Unicode planes 1-16)
  - *Two* `UniChar` values form *one* character: 𪜀
- Composing sequences (multiple characters)
  - Combining marks, clusters, ligatures, jamos
- Different “spellings” (precomposed vs. decomposed):
  뱴 ⇔ ㅂ ㅐ ㅆ
- Bidirectionality, Rearrangement
  - Glyph order differs radically from character order
  - Affects style run order, not just glyph order in run

Avoiding the Gotchas

- Use System APIs to find text boundaries
  - Character, cluster, word, line, paragraph
  - Cocoa: `NSString rangeOfComposedCharacterSeqenceAtIndex:`
  - Carbon: `CFStringGetRangeOfComposedCharactersAtIndex`, `UCFindTextBreak`
  - Truncation: `kATSULineTruncationTag`

Avoiding the Gotchas

- Use System APIs to compare or search text
  - Cocoa: `NSString compare:`
  - Carbon: `CFStringCompare, UCCompareText`
- Use System APIs to draw *entire* paragraph
  - Cocoa: `NSAttributedString, NSTypesetter`
  - Carbon: `ATSUTextLayout, ATSUStyle`
- Use System APIs to map/change cursor positions
  - Cocoa: `NSLayoutManager`
  - Carbon: `ATSUOffsetToPosition, ATSUPositionToOffset, ATSUxxxCursorPosition`

Handling Non-Unicode Text

- Use `NSString`, `CFString`, TEC to convert
- Which encoding to use?
  - `GetApplicationTextEncoding()` usually matches app HI
  - `CFStringGetSystemEncoding()` usually matches user's most-preferred language
    - Both may return a conversion-only encoding (Việt), or meaningless results (‘ōlelo Hawai‘i)
  - Internet apps should use standard IANA encodings, not Mac OS encodings

CF Dates, Times, and Numbers

- New `CFLocale` API set
  - Get current locale, or locale from ISO string
  - Maps from WorldScript world to ISO strings
- Dates, times, numbers in CF: new for Panther
  - Both formatting and parsing; includes currencies
    - `CFDateFormatter`: `CFDate`, `CFAbsoluteTime` ⇔ `CFString`
    - `CFNumberFormatter`: `CFNumber`, C type ⇔ `CFString`
  - Can customize format strings, options

Supporting TSM Directly

- `NewTSMDocument` with type ‘udoc’
- Handle CarbonEvents:
  - Unicode input
  - Active input area
  - New: whole-document content access

Carbon Events (I)

- Unicode input
  - `kEventTextInputUnicodeText`
  - Glyph variant info may be included (more later)
- Input Method active area
  - `kEventTextInputUpdateActiveInputArea`
  - `kEventTextInputOffsetToPos`
  - `kEventTextInputPosToOffset`
  - `kEventTextInputGetSelectedText`

Carbon Events (II)

- Whole-document access: model document as `CFString`
- `kEventClassTSMDocumentAccess`
  - `kEventTSMDocumentAccessGetLength`
  - `kEventTSMDocumentAccessGetSelectedRange`
  - `kEventTSMDocumentAccessGetCharactersPtr`
  - `kEventTSMDocumentAccessGetCharactersPtrForLargestBuffer`
  - `kEventTSMDocumentAccessGetCharacters`
  - `kEventTSMDocumentAccessGetFont`
  - `kEventTSMDocumentAccessGetGlyphInfo`
  - `kEventTSMDocumentAccessLockDocument`
  - `kEventTSMDocumentAccessUnlockDocument`

Supporting Variant Glyph Access

- Optional parameter: `kEventTextInputUnicodeEvent`: `kEventParamTextInputGlyphInfo`
- Array of glyph info records (`TSMGlyphInfo`):
  - Range of text for this glyph (can be >1 `UniChar`)
  - Font to use
  - Collection (e.g., AJ 1-5, or font-specific ID)
  - Glyph ID within collection
- ATSUI provides ATSUStyle tag to add to text:
  - `kATSUGlyphSelectorTag`, `ATSUGlyphSelector`
- TN 2079

Supporting Text Attributes

- Support the font panel
- Same CarbonEvent as Jaguar (`kEventFontSelection`)
- New: dict from `NSFontPanel` (`kEventParamDictionary`)
- Can contain:
  - Font features: `FontFeatureTypes`, `FontFeatureSelectors`
  - Style attributes: `FontAttributes`
    - Dictionary: `StyleAttributeTags`, `…Sizes`, `…Values`
  - Variation axes: `FontVariationAxes`, `FontVariationValues`
- Ready to pass directly to ATSUI

Demo

Yasuo Kida
Manager, OS Engineering Tokyo
International and Text Group

http://developer.apple.com/wwdc2003/urls.html

Summary

- WorldScript and QuickDraw Text are obsolete, even in major markets such as Japan
- Apple is focusing exclusively on Unicode
- Unicode support is a major competitive advantage for your application in Japan and China…
- …and gives you the rest of the world

Wrap Up

Roadmap

406 What You Need to Know About Fonts
Learn about the font panel and typography panel

Mission
Tues., 3:30pm

427 Cocoa Text
Learn about new features for Panther

Presidio
Fri., 5:00pm

428 Understanding Ink APIs
Enhance your app's support for handwriting

Nob Hill
Fri., 5:00pm

FF013 International Technologies
Your chance to give feedback

North Beach
Fri., 10:30am

Whom to Contact

Xavier Legros
International Technologies Relationship Manager
xavier@apple.com

Deborah Goldsmith
Manager, Fonts / Unicode Liaison
goldsmit@apple.com

http://developer.apple.com/wwdc2003/urls.html

For More Information

Reference Library

Documentation

International Technologies
http://developer.apple.com/intl/

Font Tools and Specifications
http://developer.apple.com/fonts/

Cocoa Application Kit Reference
ADC Home > Documentation > Cocoa > Application Kit Reference

Apple Type Services for Unicode Imaging (ATSUI)
ADC Home > Documentation > Carbon > Text and
Other International Services

Reference Library

Documentation (continued)

Unicode Utilities

ADC Home > Documentation > Carbon > Text and Other International Services > Unicode Utilities

Cocoa Text Handling Programming Topics

ADC Home > Documentation > Cocoa > Text Handling

CFString

ADC Home > Documentation > Core Foundation > Data Management > Strings

Reference Library

Technical Notes, Sample Code

TN2056: “Installable Keyboard Layouts”
ADC Home > Developer Documentation > Technical Notes

TN2079: “Glyph Access Protocol”
ADC Home > Developer Documentation > Technical Notes

TypeServicesForUnicode
ADC Home > Sample Code > Text > TypeServicesForUnicode

BasicInputMethod
ADC Home > Sample Code > Interapplication Comm > BasicInputMethod

Other Resources

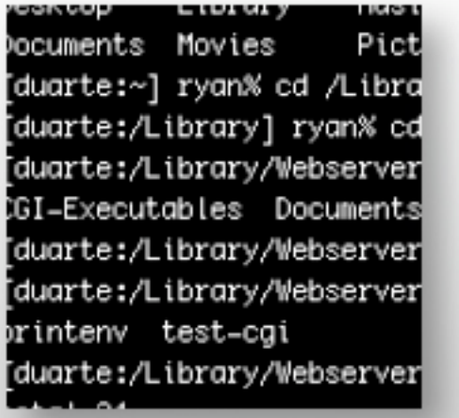
The Unicode Consortium
http://www.unicode.org/

The Unicode Standard, Version 4.0 **(forthcoming)**
Addison-Wesley, ISBN 0-321-18578-1

Unicode Demystified **by Richard Gillam**
Addison-Wesley, ISBN 0-201-70052-2

International Components for Unicode
http://oss.software.ibm.com/icu/

Q&A



Xavier Legros
International Technologies Relationship Manager
xavier@apple.com

http://developer.apple.com/wwdc2003/urls.html

WWDC 2003

Worldwide Developers Conference